Algorithm for Financial Derivatives Evaluation in a Generalized Multi-Heston Model

Daniel Negură
ERCEA Bruxelles, Belgium
dnegura@gmail.com

Abstract
In this paper we show how could a financial derivative be estimated based on an assumed Multi-Heston model support.

Keywords
Euler Maruyama discretization method, Monte Carlo simulation, Heston model, Double-Heston model, Multi-Heston model

1. Multi-Heston model for an asset

The model defined by Heston (see [1]) consists from two stochastic differential equations for a traded asset. The Heston model corrects some inconsistencies of the Black–Scholes–Merton model (see [2]), for example (see [3]):

a) in reality, volatility is not a static parameter; it can be used as a static value only for short periods (this value can be obtained based on the calibration process, usually with a statistical estimator);

b) for longer periods, it is possible that interest rate series does not verify a normal law.

The (Generalized) Heston model is described by the following coupled stochastic differential equations:

\[\begin{align*}
    dS(t) &= A(S(t), v(t), t) \, dt + B(S(t), v(t), t) \, dB_1(t) \\
    dv(t) &= C(S(t), v(t), t) \, dt + D(S(t), v(t), t) \, dB_2(t) \\
    dB_1(t) \, dB_2(t) &= r \, dt
\end{align*}\]

The (Generalized) Double-Heston model extends the volatility component to 2 semi-volatilities (see [4]):

\[\begin{align*}
    dS(t) &= M(S(t), v_1(t), v_2(t), t) \, dt + S_1(S(t), v_1(t), t) \, dB_1(t) + S_2(S(t), v_2(t), t) \, dB_2(t) \\
    dv_1(t) &= C_1(S(t), v_1(t), t) \, dt + D_1(S(t), v_1(t), t) \, dB_3(t) \\
    dv_2(t) &= C_2(S(t), v_2(t), t) \, dt + D_2(S(t), v_2(t), t) \, dB_4(t) \\
    dB_1(t) \, dB_3(t) &= r_1 \, dt \\
    dB_2(t) \, dB_4(t) &= r_2 \, dt
\end{align*}\]

This model can be extended to a multi-volatility model, called The (Generalized) Multi-Heston model:

\[\begin{align*}
    dS(t) &= M(S(t), v_1(t), v_2(t), \ldots, v_n(t), t) \, dt + S_1(S(t), v_1(t), t) \, dB_1(t) + S_2(S(t), v_2(t), t) \, dB_2(t) + \ldots + S_n(S(t), v_n(t), t) \, dB_n(t) \\
    dv_1(t) &= C_1(S(t), v_1(t), t) \, dt + D_1(S(t), v_1(t), t) \, dA_1(t) \\
    dv_2(t) &= C_2(S(t), v_2(t), t) \, dt + D_2(S(t), v_2(t), t) \, dA_2(t) \\
    \ldots \\
    dv_n(t) &= C_n(S(t), v_n(t), t) \, dt + D_n(S(t), v_n(t), t) \, dA_n(t) \\
    dB_1(t) \, dA_1(t) &= r_1 \, dt \\
    dB_2(t) \, dA_2(t) &= r_2 \, dt
\end{align*}\]
2. Financial derivative based on a support (see [3])

Any financial derivative based on a support with the price $S(t)$ at the moment $t$, with a quotation depending on the value of the support and of some volatility parameters at the moment $t$ is $V(S, v_1, v_2, \ldots, v_n, t)$, where:

$$V: R \times [0,T] \times [0,s_{1\text{max}}] \times [0,s_{2\text{max}}] \times \ldots \times [0,s_{n\text{max}}] \rightarrow R$$

and at the maturity moment $T$ it will generate a payoff:

$$\text{payoff: } R^+ \rightarrow R^+$$

For example, the European options CALL and PUT have the following payoff functions:

$$\text{payoff (x)} = \max\{0, x - E\}$$
$$\text{payoff (x)} = \max\{0, E - x\}$$

where $E$ is the exercise price of the option.

4. Path trace simulation building algorithm

We shall build a trace using the Euler-Maruyama method, after the discretization of the continuous time:

$$t[k] = t[0] + k\Delta, \ 0 \leq k \leq N$$

where:

a) $\Delta$ is the time frame dimension;

b) $N$ is the number of time units, such as days or hours;

c) $T$ is the maturity moment for the derivative.

Like (17) in [3], we have the discretization step:

$$M := M(S[k], \nu[1][k], \nu[2][k], \ldots, \nu[n][k], t[k])$$

For $i:=1$ to $n$ do

$S[i] := S_i(S[k], \nu[i][k], t[k])$

$C[i] := C_i(S[k], \nu[i][k], t[k])$

$D[i] := D_i(S[k], \nu[i][k], t[k])$

End For

$S[k+1] := S[k] + M \ast \Delta$

For $i:=1$ to $n$ do

$S[k+1] := S[k+1] + S[i] \ast X[i] \ast \sqrt{\Delta}$

$V[i][k+1] := V[i][k] + C[i] \ast \Delta + D[i] \ast Y[i]$

End For

End Algorithm
where X[i] and Y[i] are r[i]-correlated normal rand values, generated in a pre-step, such as (18) in [3]:

**Algorithm Prestep**

For i:=1 to n do
    X[i] := NormRand()
    Z := NormRand()
    Y[i] := r[i]*X[i]+Z*sqrt(1-r*r)
EndFor
EndAlgorithm

A complete path simulation can be built after assembling these algorithms into:

**Algorithm Simulation**

For i:=1 to N do
    Prestep
    Step(i)
EndFor
Simulation := payoff(S[n])
EndAlgorithm

3. Monte Carlo simulations

We can use Monte Carlo simulation to build a complexity O(1/sqrt(M)) approximation of the payoff(E[S[n]]):

**Algorithm MonteCarlo**

s := 0
for i:=1 to M do
    s := s+Simulation()
EndFor
MonteCarlo := payoff(s/n)
EndAlgorithm

This algorithm can easily be spliced on a PRAM platform like:

**Algorithm MonteCarlo**

s := 0
PARALLEL for i:=1 to M do
    s := s+Simulation()
EndFor
MonteCarlo := payoff(s/n)
EndAlgorithm

or on a message-based platform like:

**Algorithm MonteCarlo**

P := processorcount()
If processorid()=0 then
    S := 0
for i:=1 to P do
recvfromany(x)
S := S + x
Endfor
MonteCarlo := payoff(s/P)
Else
S := 0
M := M/P
For i:=1 to M do
S := S + Simulation()
Endfor
Sendto(0, S/M)
Endif
EndAlgorithm

4. Further works and acknowledgment

An extension idea is to build a multi-asset model with Multi-Heston for every asset. Another idea would be to parallelize the construction of the path with parallelizing of every computing step, because the Step and Prestep algorithms are similar for every call.

I want to thank prof. Tiberiu Socaciu from The University of Suceava for helping me with the idea of a (Generalized) Multi-Heston model.

References


